Simulation Project Analysis
CSCI-4210
Operating Systems

Mitesh Kumar <kumarm4>, Jason Lam <lamj7>, William He <hew7>

March 22, 2021

Contents

1 Best Algorithm for CPU-bound vs I/O-bound Processes

Table 6 shows the behavior of each algorithm when all processes are CPU bound. *RR* is the best performing algorithm for this test as it has the lowest average wait time of 2249.39 ms and the lowest average turnaround time of 3913.57 ms.

Table 12 shows the behavior of each algorithm when the majority of processes are CPU bound and when there are some I/O bound processes intermixed. Here, the best performing algorithm is *SRT* as it had the lowest average wait time and the lowest average turnaround time. Since the second scenario is much more representative of a real case, the team argues that *SRT* is the best algorithm for CPU bound processes.

Table 1 shows the behavior of each algorithm when all processes are I/O bound. *FCFS* seems to be the best performing algorithm as it has the lowest average wait times, turnaround times, and a middle CPU utilization percentage.

Table 14 shows the behavior of each algorithm when the majority of processes are I/O bound, but a few are CPU bound. We can see that *FCFS* is not nearly as good now because some short CPU burst processes get blocked behind the sparse set of CPU bound processes. Here *SRT* seems to be the best algorithm as it has the lowest average wait time and lowest average turnaround time. Therefore, the

team argues that, as Table 14 represents a more common scenario, the *SRT* algorithm is the best suited for I/O bound processes.

2 RR Algorithm With rr_{add} set to `BEGINNING` vs `END`

Tables 4,5,9 and 10 show data from runs that have the optional flag rr_{add}`:BEGINNING`. By comparing the results for algorithm *RR* in Table 3 and Table 4, which have the same configurations except for rr_{add}`:BEGINNING`, the team found that rr_{add}`:BEGINNING` raised the average wait time and the average turn around time, and it decreased the CPU utilization. We can therefore say that rr_{add}`:END` is better here.

3 Comparison Between SJF and SRT

The preemption added to *SJF* tends to lower average waiting times and average turnaround time while lowering CPU utilization. This difference in CPU utilization is more pronounced when there are higher context switch latencies. This is because *SRT* will always have more context switching than *SJF*. Although *SRT* is fairer, the preemptions cause a drop in CPU utilization. We can see this in the results from Table 14 where we have much lower average wait times and turnaround times in *SRT* but an almost 3% worse CPU utilization.

4 Limitations of Our Simulation

1. The I/O queue is assumed to be infinitely parallel even though it wouldn't be in reality. According to our simulation, multiple processes can join the I/O waitin gstates and their I/O waiting timers can simultaneously elapse no matter how many processes are in the I/O waiting state.

2. In our simulation, we can only have one process running at a time even though modern processors will have multiple CPU cores and therefore can have multiple processes running simultaneously.

3. Our simulator can only manage up to 26 processes even though a real system may have thousands of processes running. In order to implement this for thousands of processes, we would have to optimize our print output such that we don't have to print the queue contents for each event, and optimize our readyQueue to use a heap in our *SJF* and *SRT* algorithms.

4. Our simulator doesn't have a way to configure different probability distributions for different input parameters to the process generation. For example, we might want to use an exponential distri-

bution with one particular $\lambda_1 = 0.01$ for CPU burst times, and $\lambda_2 = 0.001$ for I/O burst times to indicate that we are simulating a set of mainly I/O bound processes.

5 Priority Scheduling Algorithm of Our Own Design

The team discovered that in CPU bound scenarios, the *RR* algorithm excessively preempts processes, resulting in lower CPU utilizations. This behavior can especially be observed in Table 7. *RR* has the lowest CPU utilization and the highest amount of preemptions. Our algorithm will double the t_{cs} of the process each time it is preempted. Therefore, each process will have an individual t_{cs} assigned to it. This should allow our system to detect CPU boundedness of processes and then adjust accordingly by allowing those processes to use the CPU for an extended amount of time. This should lower the context switch latency while allowing smaller amounts of I/O bound processes to still get through. The disadvantage would be that after many preemptions, we can still get into an occasional condition where there is an I/O bound process that has several CPU bound processes ahead of it in the queue that each have very high t_{cs} values.

6 Collected Data Tables

Program Arguments

`[executable]` `[`n`]` `[seed]` `[`λ`]` `[limit]` `[`t_{cs}`]` `[`α`]` `[`t_{slice}`]` `[`rr_{add}`: BEGINNING or END, default: END]`

Program Execution Data

Table 1: I/O bound `[`n`:1]` `[seed:2]` `[`λ`:0.01]` `[limit:256]` `[`t_{cs}`:4]` `[`α`:0.5]` `[`t_{slice}`:128]`

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	286.957	82.3043	373.261	23	0	39.3185
SJF	286.957	87.5217	378.478	23	0	38.5514
SRT	286.957	87.5217	378.478	23	0	38.5514
RR	286.957	95.6522	388.87	36	13	38.1856

Table 2: I/O bound [n:16] [seed:2] [λ:0.01] [limit:256] [t_{cs}:4] [α:0.75] [t_{slice}:64]

Algorithm	Average CPU Burst Time (ms)	Average Wait Time (ms)	Average Turnaround Time (ms)	Total Number of Context Switches	Total Number of Preemptions	CPU Utilization (%)
FCFS	286.957	82.3043	373.261	23	0	39.3185
SJF	286.957	87.5217	378.478	23	0	38.5514
SRT	286.957	87.5217	378.478	23	0	38.5514
RR	286.957	93.6957	389.174	49	26	38.1062

Table 3: I/O bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:4] [α:0.5] [t_{slice}:2048]

Algorithm	Average CPU Burst Time (ms)	Average Wait Time (ms)	Average Turnaround Time (ms)	Total Number of Context Switches	Total Number of Preemptions	CPU Utilization (%)
FCFS	286.957	82.3043	373.261	23	0	39.3185
SJF	286.957	87.5217	378.478	23	0	38.5514
SRT	286.957	113.522	405.174	27	4	38.5424
RR	286.957	82.3043	373.261	23	0	39.3185

Table 4: I/O bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:4] [α:0.5] [t_{slice}:2048] [rr_{add}:BEGINNING]

Algorithm	Average CPU Burst Time (ms)	Average Wait Time (ms)	Average Turnaround Time (ms)	Total Number of Context Switches	Total Number of Preemptions	CPU Utilization (%)
FCFS	286.957	82.3043	373.261	23	0	39.3185
SJF	286.957	87.5217	378.478	23	0	38.5514
SRT	286.957	113.522	405.174	27	4	38.5424
RR	286.957	156.217	447.174	23	0	38.6281

Table 5: I/O bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:20] [α:0.5] [t_{slice}:2048] [rr_{add}:BEGINNING]

Algorithm	Average CPU Burst Time (ms)	Average Wait Time (ms)	Average Turnaround Time (ms)	Total Number of Context Switches	Total Number of Preemptions	CPU Utilization (%)
FCFS	286.957	88.4783	395.435	23	0	38.9841
SJF	286.957	95.8696	402.826	23	0	38.1944
SRT	286.957	106.522	416.087	26	3	38.1503
RR	286.957	163.87	470.826	23	0	38.3053

Table 6: CPU bound [n:1] [seed:2] [λ:0.01] [limit:256] [t_{cs}:4] [α:0.5] [t_{slice}:128]

Algorithm	Average CPU Burst Time (ms)	Average Wait Time (ms)	Average Turnaround Time (ms)	Total Number of Context Switches	Total Number of Preemptions	CPU Utilization (%)
FCFS	1629.22	3216.87	4850.09	23	0	98.0326
SJF	1629.22	2596.35	4229.57	23	0	94.7603
SRT	1629.22	2543.57	4178.17	31	8	94.6836
RR	1629.22	2249.39	3913.57	201	178	92.0236

Table 7: CPU bound [n:16] [seed:2] [λ:0.01] [limit:256] [t_{cs}:4] [α:0.75] [t_{slice}:64]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1629.22	3216.87	4850.09	23	0	98.0326
SJF	1629.22	2596.35	4229.57	23	0	94.7603
SRT	1629.22	2543.57	4178.17	31	8	94.6836
RR	1629.22	2327.13	4024.17	390	367	90.0683

Table 8: CPU bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:4] [α:0.5] [t_{slice}:2048]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1629.22	3216.87	4850.09	23	0	98.0326
SJF	1629.22	2730.65	4363.87	23	0	95.3146
SRT	1629.22	2017.39	3652.35	33	10	94.6645
RR	1629.22	2829.35	4463.78	30	7	95.2468

Table 9: CPU bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:4] [α:0.5] [t_{slice}:2048] [rr_{add}:BEGINNING]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1629.22	3216.87	4850.09	23	0	98.0326
SJF	1629.22	2730.65	4363.87	23	0	95.3146
SRT	1629.22	2017.39	3652.35	33	10	94.6645
RR	1629.22	1743.61	3378.04	30	7	95.6162

Table 10: CPU bound [n:8] [seed:64] [λ:0.001] [limit:4096] [t_{cs}:20] [α:0.5] [t_{slice}:2048] [rr_{add}:BEGINNING]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1629.22	3255.13	4904.35	23	0	97.0978
SJF	1629.22	2761.26	4410.48	23	0	94.4307
SRT	1629.22	2057.74	3715.65	33	10	93.4184
RR	1629.22	2876.65	4531.96	30	7	94.0987

Table 11: Mostly CPU bound [n: 8] [seed: 64] [λ: 0.001] [limit: 4096] [t_{cs}: 20] [α: 0.5] [t_{slice}: 2048]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1256.77	3088.9	4365.67	30	0	63.1108
SJF	1256.77	2744.7	4021.47	30	0	61.9799
SRT	1256.77	1960.37	3245.13	42	12	70.4413
RR	1256.77	2608.07	3890.17	38	8	69.7958

Table 12: Mostly CPU bound [n: 8] [seed: 64] [λ: 0.001] [limit: 4096] [t_{cs}: 4] [α: 0.5] [t_{slice}: 2048]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1256.77	3041.43	4302.2	30	0	63.6219
SJF	1256.77	2703.63	3964.4	30	0	62.4729
SRT	1256.77	2161.47	3423.57	40	10	62.1946
RR	1256.77	2549.93	3811.77	38	8	70.5692

Table 13: Mostly I/O bound [n: 8] [seed: 64] [λ: 0.001] [limit: 4096] [t_{cs}: 20] [α: 0.5] [t_{slice}: 2048]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1256.77	3088.9	4365.67	30	0	63.1108
SJF	1256.77	2744.7	4021.47	30	0	61.9799
SRT	1256.77	1960.37	3245.13	42	12	70.4413
RR	1256.77	2608.07	3890.17	38	8	69.7958

Table 14: Mostly I/O bound [n: 8] [seed: 64] [λ: 0.001] [limit: 4096] [t_{cs}: 4] [α: 0.5] [t_{slice}: 2048]

Algorithm	**Average CPU Burst Time (ms)**	**Average Wait Time (ms)**	**Average Turnaround Time (ms)**	**Total Number of Context Switches**	**Total Number of Preemptions**	**CPU Utilization (%)**
FCFS	1710.3	3412.33	5126.63	30	0	99.4187
SJF	1710.3	2505.97	4220.27	30	0	99.1842
SRT	1710.3	756	2472.03	43	13	96.731
RR	1710.3	1314.93	3030.17	37	7	96.8295